Scilab/Scicos: an Open Source Platform for Embedded Real Time Systems Development

Claude Gomez, Simone Mannori

Scilab/Scicos: an Open Source Platform for Embedded Real Time Systems Development

Claude GOMEZ[1], Simone MANNORI[2]

1: CTO, INRIA/Scilab Consortium (claude.gomez@inria.fr)
2: Embedded System Engineer, INRIA/Scilab Consortium (simone.mannori@inria.fr)
INRIA/Scilab Consortium, Domaine de Voluceau, B.P. 105, 78153 Le Chesnay, France

Abstract: Complex, heterogeneous, real time embedded applications require sophisticated developments tools for simulation and implementation. Commercial, closed source applications are available "on the shelf", but the associated financial effort and the limited flexibility are not always compatible with the budget constraints and the technical specifications. Some specific requirements of the embedded applications match very well the intrinsic proprieties of the Open Source software. In this paper we show how Scilab/Scicos can play the role of an Open Source platform for embedded systems by presenting both its development model and its applications.

Keywords: Scilab, Scicos, embedded systems, real time, multi platform, Open Source scientific software.

1. Introduction

The purpose of this paper is to present how the Open Source platform for numerical computation Scilab/Scicos [1] can be used for embedded real time systems. But as well as presenting the functionalities, we also explain why it is important that this platform be an Open Source platform and which is the corresponding organization of Scilab development.

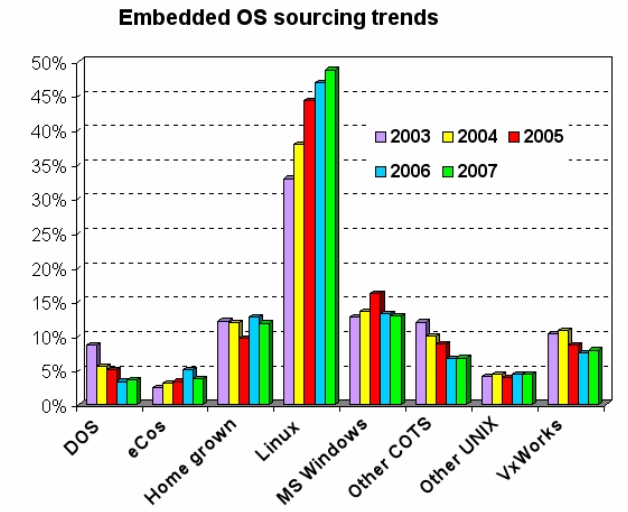


Figure 1: Open Source trends for embedded.

2. Why is Open Source important for embedded real time applications?

Today more and more Open Source software is used for industrial applications [2] (fig.1). This is a new economical model where software is given and business is made by service and support. This has well known advantages for users such as the independence with respect to the vendor, the reduction of recurring costs, the flexibility of use, etc…
In the domain of embedded real time applications, there are more arguments for Open Source software:

- **Safety.** For safety critical applications, the full access to the source code (of the tools and the target's code) is indispensable. Every certification needs code inspection. The release of the code under NDA agreements is not sufficient because of the impossibility of independent evaluations.
- **Reduce non recurring costs.** The acquisition of proprietary tools is normally associated to high initial costs. In Europe, most of the innovative development is done by small companies with modest budget that cannot sustain this high initial investment.
- **Reduce recurring costs.** Proprietary tools need frequent updates. These update are available only after annual subscription. The code produced by proprietary tools is frequently associated to royalties.
- **Performances.** Using Open Source tools, it is possible to benchmark tools and target code keeping one eye on the sources. Developers are free to modify the code and experiment alternative solutions. The inspections and the consequent optimization, if required, can be pushed to assembler level.

But the development of Open Source software must be made in a professional way in order to be used by both Academics and Companies, leaving the possibility to be helped by the community of contributors. This is what we show in the following sections.

3. Scilab until 2007. Development and Structure: the Scilab Consortium

3.1 Short history of Scilab: why Open Source?

The origin of Scilab comes from the 80's and was based upon the original Matlab software written in FORTRAN code. It was first sold by a subsidiary of INRIA and in the 90's it comes back to INRIA. Then it was developed by researchers from INRIA and ENPC and it was decided to be distributed freely on the Internet in 1994.

Numerical computations are used in a large variety of strategic domains (defense, aerospace, energy, communications, etc.) and there is nearly only one such kind of software: a proprietary tool and a "de-facto monopoly. So there is a **need for Free Open Source Software in Numerical Computation** and it was the reason of the decision to distribute Scilab as free software.

3.2 Towards professional Open Source numerical computation software: the Scilab Consortium

Regarding the success of this operation, mainly in the academic domain, INRIA decided to create a Consortium to initiate the transfer of Scilab into professional software. The purpose was to build a structure (fig.2) and a dedicated team for taking care of Scilab in order to boost the use of Scilab both by Academics and by Companies.

Scilab Consortium was created in 2003 and hosted by INRIA. The team was created as a development team at INRIA. Today there are 25 members in Scilab Consortium: ANAGRAM TECHNOLOGIES, APPEDGE, ARTENUM, AXS INGENIERIE, ATMEL ROMA, ENGNET, CEA, CNES, DASSAULT AVIATION, ECOLE CENTRALE DE PARIS, ECOLE POLYTECHNIQUE, EADS, EDF, ENPC, ESTEREL TECHNOLOGIES, IFP, INRIA, KLIPPEL, MANDRIVA, PSA PEUGEOT CITROËN, RENAULT, SCALEO CHIP, STYREL TECHNOLOGIES, THALES and TNI.

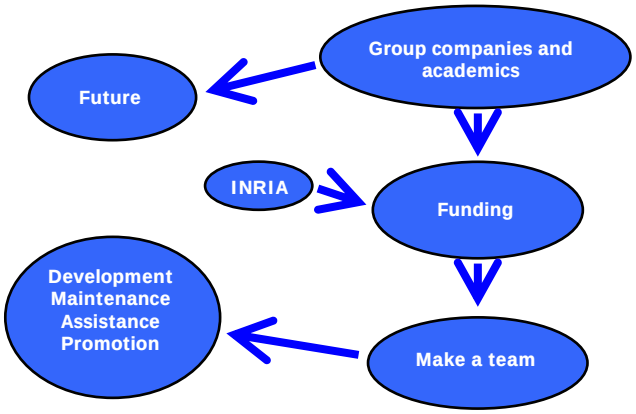


Figure 2: Scilab Consortium model.

Since 2003, Scilab Consortium and the dedicated team have made Scilab versions and Scilab promotion in the world. The latest Scilab version is Scilab 4.1.2 released October 2007. It is now professional software (fig.3).

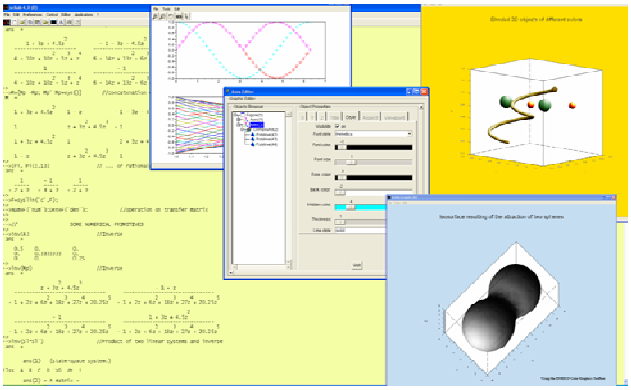


Figure 3: Scilab in action.

Scilab is used all around the world and there are now about 50,000 monthly downloads from Scilab web site.

4. Scilab after 2007: Scilab 5 and the non profit organization

We can consider that the 5-year INRIA-hosted Scilab Consortium was the first step for transferring Scilab software made by researchers to Scilab software made by development dedicated team. Now we are starting a second step by two major actions.

4.1 New Scilab software

Next Scilab release, Scilab 5, will be available at the end of March 2008. It is the beginning of a new Scilab family and it can be considered as a technological leap:

- Complete cleaning and reorganization of source code.
- New GUI and graphics rendering with Java technology.
- Modularization.

Modularization is a very important improvement (fig.4). First it allows easier collaboration with the contributors. Second it allows changing the license: Scilab 5 will be GPL compatible by adopting CeCILL license [3].

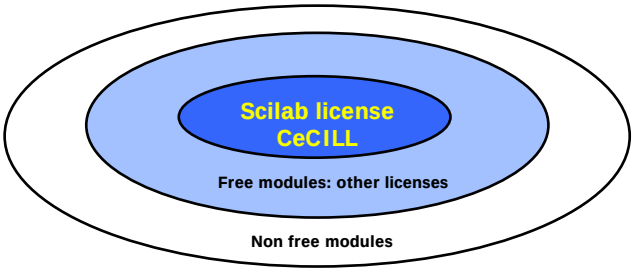


Figure 4: Scilab 5 modular architecture.

So Scilab 5 will be Free Software according to the definition of the Free Software Foundation.

It will allow Scilab to be really open software ready for use and collaboration with a big community of users both in Academic world and in Industrial world.

4.2 New Scilab structure

Together with new Scilab 5.0, a new organization will take place for Scilab. New Scilab Consortium will be a non-profit organization, following the example of Mozilla Foundation. New Scilab Consortium will be completely dedicated to Scilab and in charge of Scilab development, promotion and support for the members of the organization.

The new organization will be set up before mid 2008 and will play the role of real software publisher.

5. Generating code from Scilab

Scilab itself can be used as a tool for embedded systems development. This work is done in the European integrated project named “hArtes” funded by FP6 call 5 Embedded Systems [4].
The aim of the project is to build a heterogeneous development platform for embedded system. The complexity of future real-time embedded systems for consumer and professional products is becoming too big to design monolithic processing platforms. “Monolithic” means based on single hardware and software architecture (e.g. micro-controller with C compiled code). “Heterogeneous” means based on different, mixed and interconnected, hardware and software architecture like DSP, micro-controllers, FPGA, ASIC, general purpose processor, etc. that requires different programming languages and development tools.

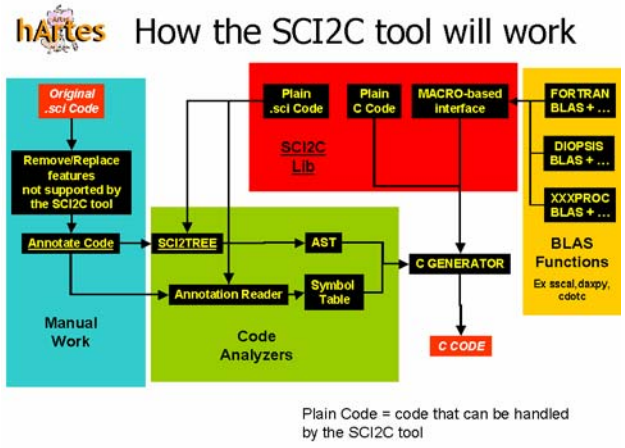


Figure 5: From Scilab script to C code.

Scilab Language is adopted as one of the starting points of the project for C code generation.
For that we are reorganizing the Scilab computational kernel in order to support “visitors oriented” code generation tools.

The work is to create a Scilab to C translator that will generate minimal C code from Scilab scripts (fig. 5). Then it will be possible to put the C generated code in processors such as FPGA or DSP.
So Scilab interpreter can be used to easily create the program and then efficient C code is automatically generated.

6. Scilab/Scicos for Embedded Systems

Scilab-EMB: (Scilab Embedded)

For some complex control applications the presence on the target of the full Scilab/Scicos installation is a reasonable solution. Modern Industrial PCs are fast enough to run Scilab/Scicos in real time but the associated costs and technical limitations (power, heath, etc.) can render this solution inappropriate.
Scilab-EMB [5] is the implementation of a full, optimized Linux and Scilab/Scicos integrated installation on an industrial PC based on ARM processor. The ARM processor has sufficient performances at reduced costs with minimal power requirements (50% cost reduction respect a standard industrial PC, no heat sinks, no fans).

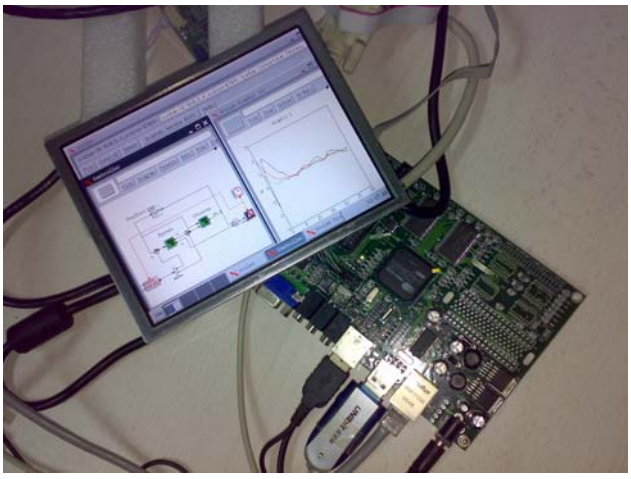


Figure 5: Scilab EMB.

The full documentation of the project it is available on the web site [5].
The porting procedure is fully based on multi platform open source tools. With minimal effort, it is possible to use the porting guide for other hardware architectures (PPC, MIPS, OMAP, etc.).
For hard real time application, a patched Linux kernel version will be available.
The use of a platform different from x86 offer another advantage: improved immunity from virus.

Scicos-EMB is developed by professor Ma Longhua and Zhe Peng the Zhejiang University (China).

7. Scicos for Embedded Systems

7.1 Scicos

Scicos [6] [7] is a Scilab toolbox for modeling and simulation of dynamical systems where continuous-time and discrete-time components are interconnected. Implementing these simulations using only Scilab scripts is possible but is complex and time consuming. Scicos uses a graphical editor where the user can place blocks and connect it with links: the usual "block diagram" control system representation (fig. 6).

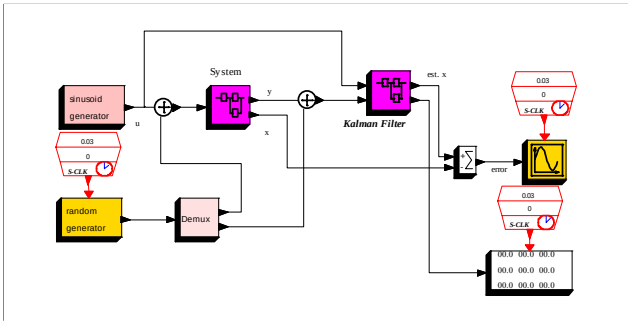
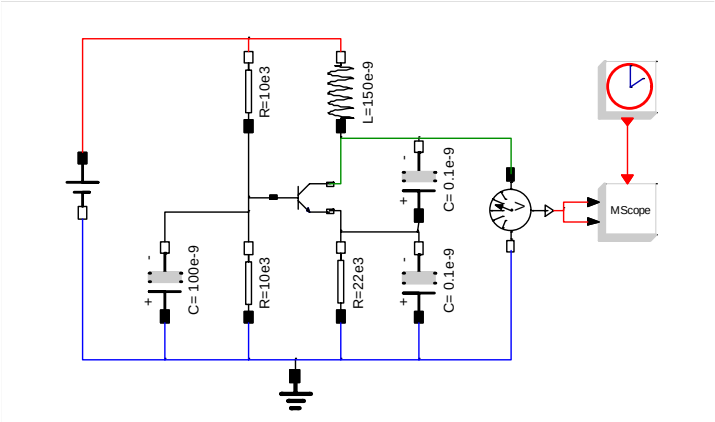


Figure 6: Typical Scicos diagram.

Scicos includes palettes of standard blocks that cover most of the usual situations found in control applications. The user is free to assemble standard blocks to create custom blocks or write directly the code. The block's code is composed by an "interfacing function" and a "computational/simulation function". The former is used by the editor to interact with the block's internal parameters; the latter is used by the Scicos simulator. Usually, for optimal performances and maximum flexibility, the computational function is written in C code and compiled. The Scilab language is supported for early development, but not suggested for performance reasons (Scilab language is interpreted, slower that compiled C functions). FORTRAN is supported but not suggested for new development. Other language can be easily used if required (e.g. Java using JNI).



The code generation [9] was initially developed to accelerate the simulation of complex super-blocks (hierarchical user diagrams composed by many blocks). The code generator integrates the information of the Scicos compiler (that decides the sequence of the calls to each computational function) and the computational function's source code to create a single source file. This file becomes the super-block's computational function.

This kind of code generation can be used, with some manual effort, for embedded real time target.

8. Scicos Toolboxes for Embedded Real Time Systems

Modern software simulation tools have changed radically the development flow of the control for embedded system engineers. Scilab and Scicos can guide the designer to elaborate and optimize complex control strategies but the results must be implemented and tested on real systems. This means doing some tests using the simulator connected to the real – physical - system and producing code to be run on the embedded target. There are several specialized toolboxes in Scicos targeting different specific needs:

- Scicos-HIL (Hardware In the Loop) allows the testing and validation of models and controllers inside the interactive, graphical, Scicos environment.
- Scicos-RTAI is a specific code generator capable to produce standalone hard real time executables under Linux RTAI.
- Scicos-FLEX is another customization of the Scicos code generator for small DSP and micro controller target.
- Scicos-HDL is a simulator and code generator extension for programmable logic circuits application using HDL (High Level Hardware Languages, VHDL and Verilog).

These toolboxes are described below.

8.1 Scicos-HIL

Modern PCs are fast enough to simulate complex systems in real time. The main idea behind Scicos "Hardware In the Loop" [10] is to use directly Scicos to simulate part of a control system in close connection with a real one (fig. 8).

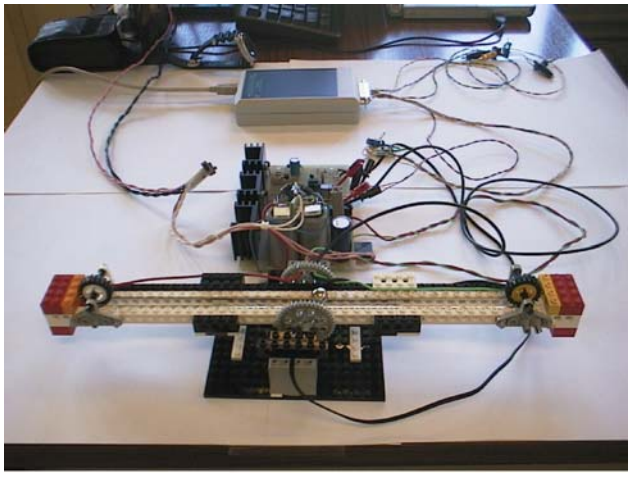
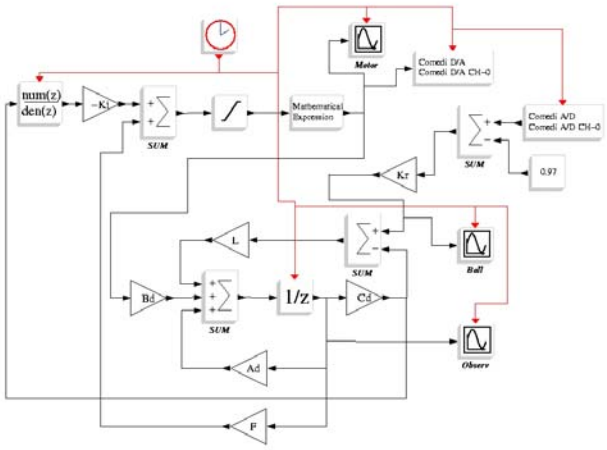


Figure 8: Ball and beam demo model.

This kind of “in the loop” simulation can be used to tune a controller or validate a model (fig.9).



only using expensive, closed source, patches.

8.1.2 Scicos-HIL on Linux

Recent Linux kernels (2.6.2x) [12] offer “near hard real time” performances for sampling time up to one millisecond. The Comedi [13] project offers a complete set of I/O drivers (more than 200 models) and a unified library. Comedi offers advanced functions for board inspection and configurations. This means that the same code can run using different boards from different constructors.

8.1.3 Scicos-HIL on Linux - RT_PREEPT

The “real time pre emption patch” [14] is not yet a Linux standard functionality, but many Linux distributions begin to offer pre-configured kernel. This patch transforms Linux in a real hard real time OS (without loose any compatibility with standard applications) and push the limit of the sampling time to 100 microseconds (real performances are configuration dependent). Probably, next year this hard real time extension will be included in the Linux source code main line and will be adopted by the major distributions (IBM, Red Hat and HP are the main sponsors of this project). We provide a patch to the standard Scilab/Scicos real time functions that detects and uses the hard real time features of RT_PREEPT.

8.2 Scicos-RTAI

RTAI (Real Time Application Interface, [15]) is a hard real time extension of the standard Linux kernel. RTAI creates a special class of Linux process and threads (both in kernel and user space). The standard Linux programs continue to run in the “non hard real time” space but can be pre-empted any time by RTAI programs. Using this “dual kernel” mechanism, the maximum latency (for real time applications) is reduced to very low values: with a well tuned machine is possible to push the sampling time to 10 microseconds (the “suggested” minimum sampling time is 40 microseconds).

Scicos-RTAI is a custom version of the standard Scicos code generation that includes RTAI for hard real time support and Comedi for I/O interface. With a single click operation, a complex Scicos diagram is compiled to a stand alone Linux RTAI application. Internal parameters control and real time visualization are realized using a RTAI-Lab [16] control panel (fig. 10).

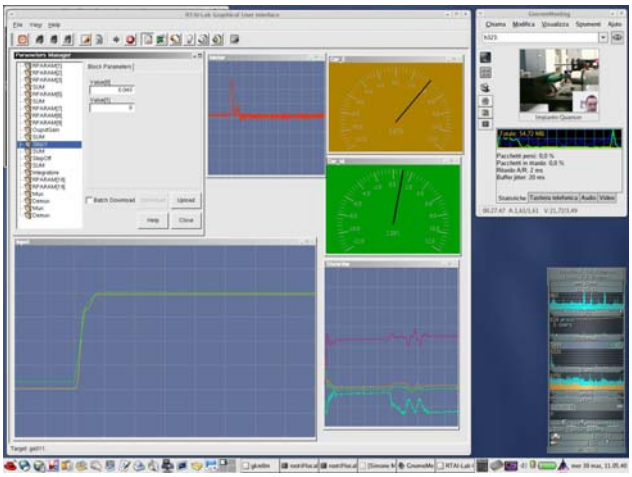


Figure 10: RTAI-Lab control panel.

RTAI Lab and the standalone executable exchange real time data using a custom UDP protocol via Ethernet network. The same Ethernet connection can be shared between RTAI applications and standard Linux applications (e.g. video conferencing, see fig.10). RTAI-Lab requires a Linux-RTAI equipped PC: RTAI-XML [17] is a client/server application that replicates the RTAI-Lab virtual instruments and control panels in a Java enabled web browser (Java RTAI Lab, fig.11).

Figure 11: JRtaiLab control panel.

Before “RT_PREEPT”, RTAI was the only hard real time open source option for Linux. RTAI is an open source project lead by Paolo Mantegazza (Politecnico di Milano, Italy). Scicos-RTAI is a project developed and maintained by Roberto Bucher in collaboration with the INRIA/Scilab Consortium and the RTAI Team.

8.3 Scicos-FLEX

Most of the usual Scicos embedded applications are closed loop controls: using a full PC to implement them can be a real waste of resources. A micro-controller/DSP is a better solution from many points of view, but the development of micro-controller/DSP control applications is expensive in terms of time and hardware and software tools. Recently, micro-controller and DSP silicon providers give free tools (integrated development environment, assembler, C compiler, libraries, etc.), but the coding is still very time consuming because the programmer is responsible of all the fine details like I/O management and real time support. Practically, this means that most of the development time is “lost” for the low level device management and not invested in the real customer’s application.

Scicos-FLEX [18] is the combination of:

- a custom code generator and specific palettes of Scicos blocks
- a real time OS optimized for micro-controllers (ERIKA)
- an integrated environment (RT-DRUID) based on Eclipse
- a development board.

The control loop are designed and simulated with Scilab/Scicos; when the simulation is ready, the code is automatically generated, compiled, linked with the operating system and downloaded on the target board via USB connection (fig 12).

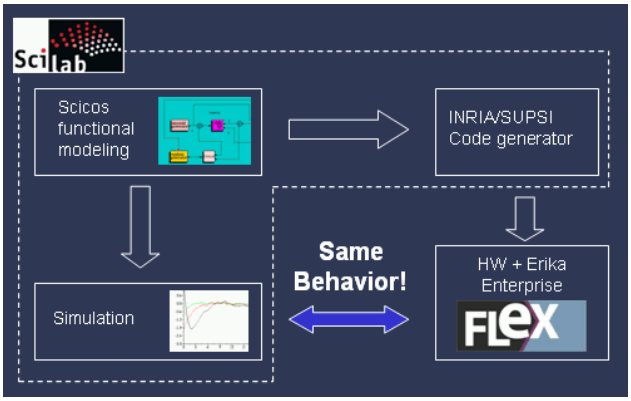


Figure 12: Scicos-FLEX development flow.

The same USB port can be used to monitor and remotely control the internal parameters of the running controller on the target (fig. 13).

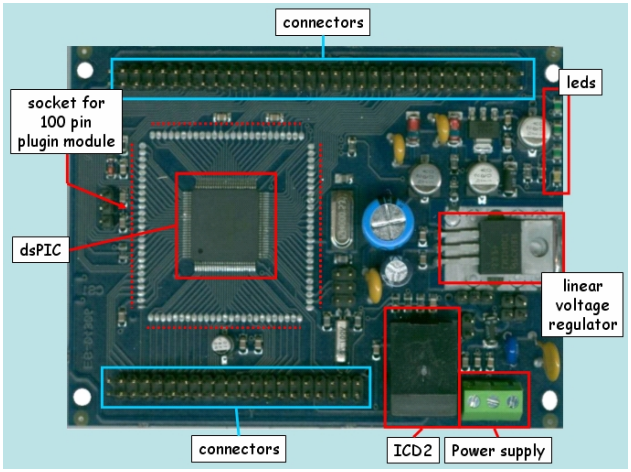


Figure 13: Scicos-FLEX (light) board.

Scicos-FLEX is a joint development of the INRIA/Scilab Consortium, Roberto Bucher (code generation tools) and Evidence (real time OS, integrated development environment, hardware support).

8.4 Scicos-HDL

The latest FPGA devices are sufficiently rich to implement, at cost comparable to “hard-coded” solutions, very complex functions. This possibility opens the door to real “system on chip” realizations where there is only one chip in the system that implements all the functions. Some FPGA are capable to integrate also analog / mixed signal functions (linear amplifier, data converter, etc.).

Silicon providers offer free (but not open source) tools for the device configuration (using high level definition languages such as Verilog or VHDL) and digital only simulation tools: these tools are not capable to simulate the full system.

Scicos-HDL [19] is an extension for Scilab/Scicos to the world of digital design. It includes palettes of the standard combinatorial and sequential digital basic

blocks, a palette of IP-CORE blocks and a Scicos-HDL specific palette of tools (fig. 13).

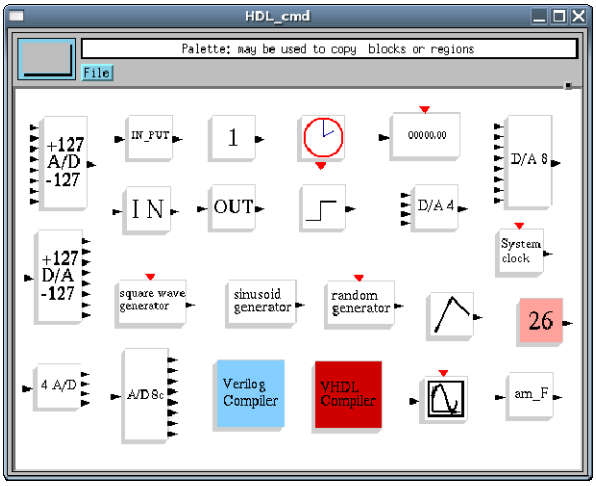


Figure 13: Scicos-HDL main palette.

Using Scicos-HDL is possible simulate the full analog, mixed-signal, digital system. Scicos-HDL includes an automatic code generation tool capable to produce Verilog or VHDL code for the “digital only part” of the simulation (fig.14).

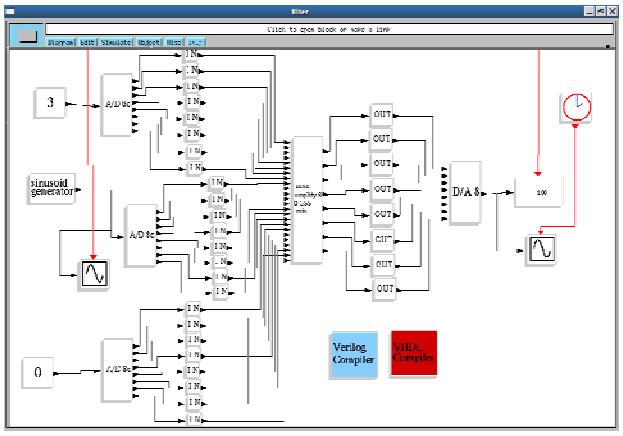


Figure 14: Scicos-HDL diagram.

This code can be “cut and pasted” inside the compiler that produces binary configuration file for the physical devices.

In the actual release (0.50), Scicos-HDL is capable to simulate at reasonable speed only small/medium complexity circuits. For the simulation phase, standard Scicos functions written in Scilab or C language are used. These simulation functions are not optimized for speed. For next releases we are working on the integration of the GHDL compiler inside the Scicos simulator. GHDL [20] is a native VHDL compiler based on the GNU-GCC structure. GHDL is capable to translate VHDL code in an optimized shared library usable for simulation. This kind of integration (still in development) opens the way to multithread co-simulation inside Scicos, with evident advantages of speed on multi cores and multi processors PC. We are integrating the GHDL produced shared libraries as separate, independent threads using semaphores for the synchronization.

Scicos-HDL is developed by Zhang Dong and Kang Cai (Ningxia University, LIAMA, China Institute of Automation, Chinese Academy of Sciences) in close collaboration with the INRIA/Scilab Consortium.

9. Conclusion

The Open Source software can be a valid alternative to closed source solution, not only for economic reasons but also for intrinsic flexibility of the development model that matches most of the technical challenges of the embedded world.

10. References

[1] www.scilab.org

[2] www.linuxdevices.com/articles/AT7065740528.html

[3] www.cecill.info/index.en.html

[4] www.hartes.org

[5] www.zjufrontech.com/scilab

[6] www.scicos.org

[7] S. L. Campbell, J-Ph. Chancelier et R. Nikoukhah: "*Modeling and Simulation in Scilab/Scicos*", Springer, 2005.

[8] www.modelica.org

[9] ref. [7], cap. 12, pag.253

[10] www.scicos.org/scicoshil.html

[11] www.measurementcomputing.com

[12] www.kernel.org

[13] www.comedi.org

[14] http://rt.wiki.kernel.org

[15] www.rtai.org

[16] RTAI-Lab pages on www.rtai.org

[17] http://artist.dsi.unifi.it/rtaixml

[18] Scicos-FLEX: http://www.evidence.eu.com/content/view/177/2

[19] Scicos-HDL: http://scicoshdl.sourceforge.net/index_en.htm

[20] GHDL: http://ghdl.free.fr